–

CIRES Technologies – Technical Challenge
Mini SOC Deployment with Wazuh & Custom SSH Detection Rule

Head of Technical Recruitment : M. Toufik AIRANE

Candidate : Soufiane SEDDIK

Position Applied For : SOC Architect / DevOps Engineer

Company : CIRES Technologies

Submission Date : September 1, 2025

Summary

This report details the successful completion of the Cires Technologies technical challenge for the SOC Architect / DevOps Engineer role. The project involved the design, implementation, and documentation of a comprehensive security solution centered on two key components.

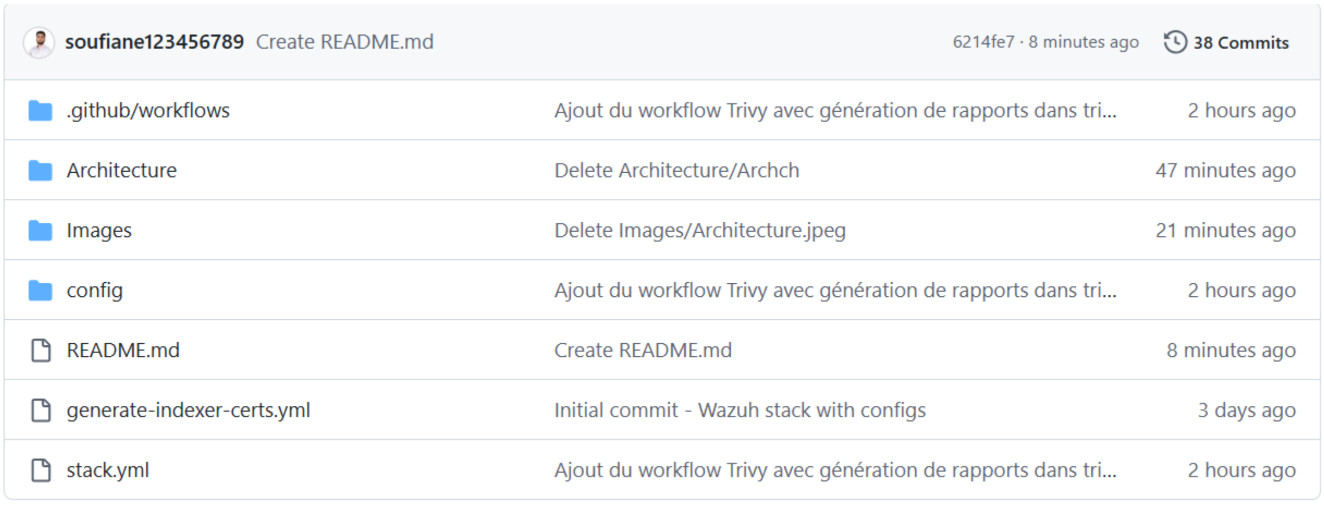
Part One focused on building a Mini SOC. A robust CI/CD pipeline, leveraging GitHub Actions, was designed to automate the deployment of a complete Wazuh stack (Manager, Indexer, and Dashboard) onto a Docker Swarm cluster. This solution ensures a scalable, secure, and reproducible environment for security monitoring.

Part Two involved developing a custom threat detection rule. A specific Wazuh rule was successfully implemented to identify suspicious SSH login patterns, specifically multiple failed attempts followed by a successful login by a new user. This rule demonstrates advanced problem-solving skills and a deep understanding of threat detection principles.

All deliverables, including the GitHub repository, architectural diagrams, and a detailed technical walkthrough, are provided to showcase the methodology and results. The project successfully meets all the technical requirements of the challenge, demonstrating strong expertise in DevOps, container orchestration, and cybersecurity.

Repository Structure

The project has been organized in a logical and intuitive manner to ensure maximum clarity and easy navigation. The repository structure is as follows:



soufiane123456789 Create README.md — 6214fe7 · 8 minutes ago — 38 Commits

.github/workflows	Ajout du workflow Trivy avec génération de rapports dans tri...	2 hours ago
Architecture	Delete Architecture/Archch	47 minutes ago
Images	Delete Images/Architecture.jpeg	21 minutes ago
config	Ajout du workflow Trivy avec génération de rapports dans tri...	2 hours ago
README.md	Create README.md	8 minutes ago
generate-indexer-certs.yml	Initial commit - Wazuh stack with configs	3 days ago
stack.yml	Ajout du workflow Trivy avec génération de rapports dans tri...	2 hours ago

This repository is logically organized to facilitate understanding and navigation of the project. The config folder centralizes all necessary configuration files, while the .github/workflows directory contains the CI/CD pipeline configurations. The Architecture and Images folders provide the conceptual diagrams and visual proofs of the project's successful execution, respectively. This structure allows for quick identification of each project component, from configuration files to architectural documents and visual proof of deployment.

Part 1: CI/CD Pipeline and Mini SOC Deployment

Introduction

The primary objective of this project's first phase was to design and implement a comprehensive CI/CD pipeline to automate the deployment of the Wazuh security information and event management (SIEM) system. This solution ensures a reliable, scalable, and automated deployment process, minimizing manual intervention and providing a consistent environment.

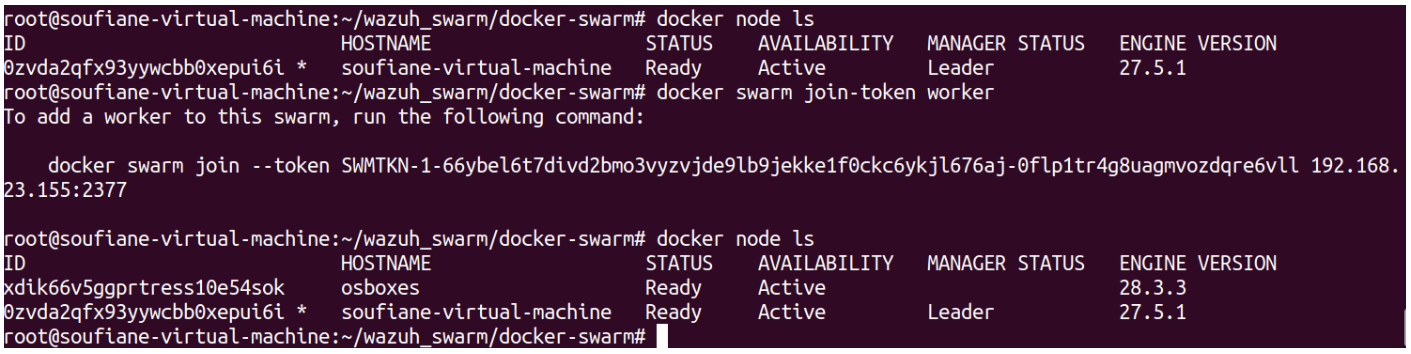
Technical Implementation

The deployment began with the creation of a Docker Swarm cluster, which serves as the foundation for the containerized environment. A dedicated worker node, named osboxes, was added to the cluster to handle the workload efficiently. Following the cluster setup, the necessary stack files were generated to define the services and their configurations.

The core of this part of the challenge was the deployment of the Wazuh cluster in a distributed architecture. This included:

- A Wazuh Manager operating in master mode.
- A Wazuh Manager configured as a worker node, ensuring high availability and load balancing.
- Three Wazuh Indexers for data synchronization and storage, providing a scalable and fault-tolerant backend.
- A Wazuh Dashboard for a centralized and intuitive visualization of security data and alerts.

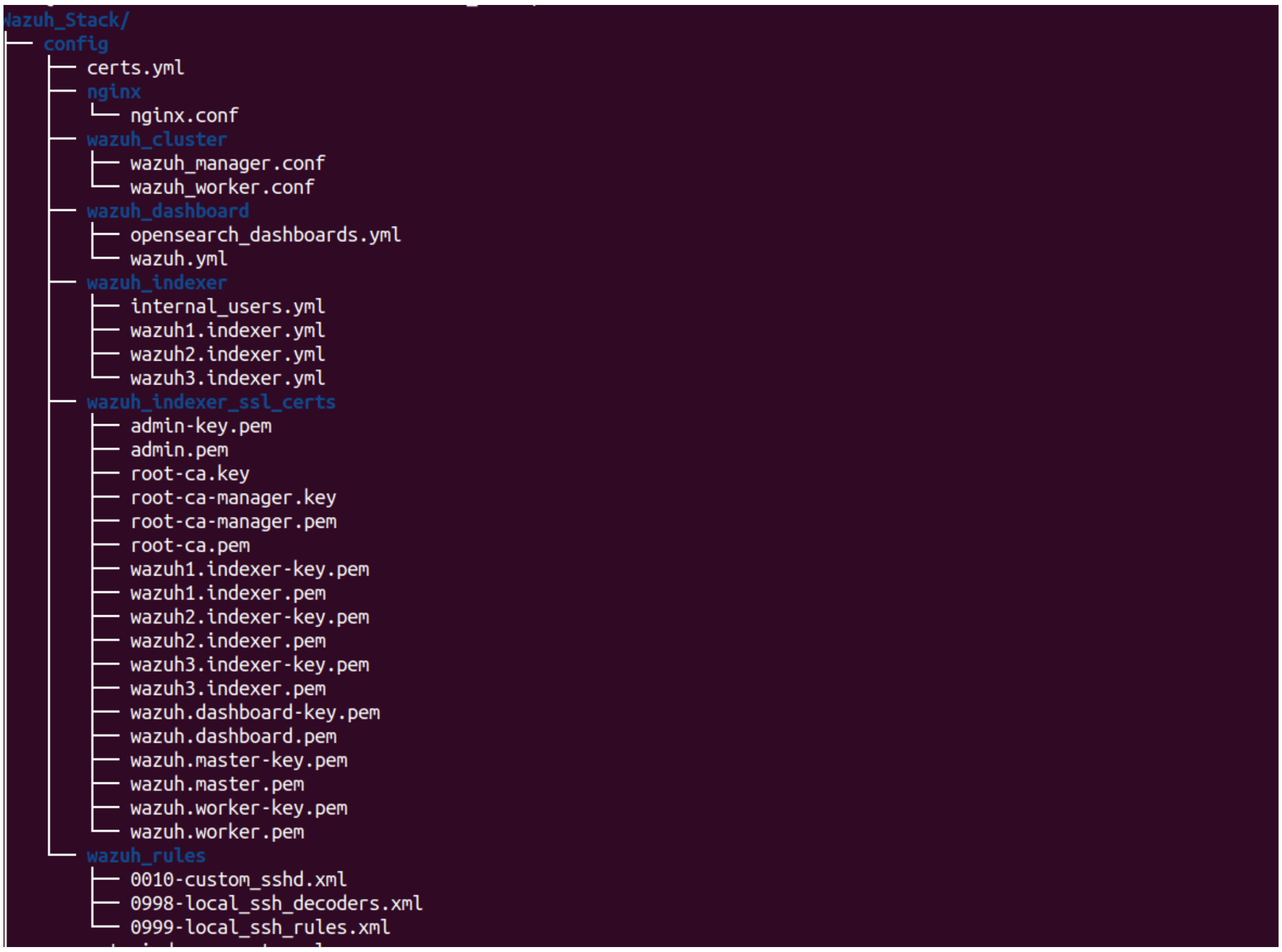
This architecture was deployed directly onto the Docker Swarm cluster, ensuring that the entire Wazuh stack runs seamlessly and securely, ready to begin monitoring.



```
root@soufiane-virtual-machine:~/wazuh_swarm/docker-swarm# docker node ls
ID                            HOSTNAME                   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
0zvda2qfx93yywcbb0xepui6i *   soufiane-virtual-machine   Ready     Active         Leader           27.5.1
root@soufiane-virtual-machine:~/wazuh_swarm/docker-swarm# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-66ybel6t7divd2bmo3vyzvjde9lb9jekke1f0ckc6ykjl676aj-0flp1tr4g8uagmvozdqre6vll 192.168.
23.155:2377

root@soufiane-virtual-machine:~/wazuh_swarm/docker-swarm# docker node ls
ID                            HOSTNAME                   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
xdik66v5ggprtress10e54sok     osboxes                    Ready     Active                          28.3.3
0zvda2qfx93yywcbb0xepui6i *   soufiane-virtual-machine   Ready     Active         Leader           27.5.1
root@soufiane-virtual-machine:~/wazuh_swarm/docker-swarm# 
```

Configuration and Stack Files

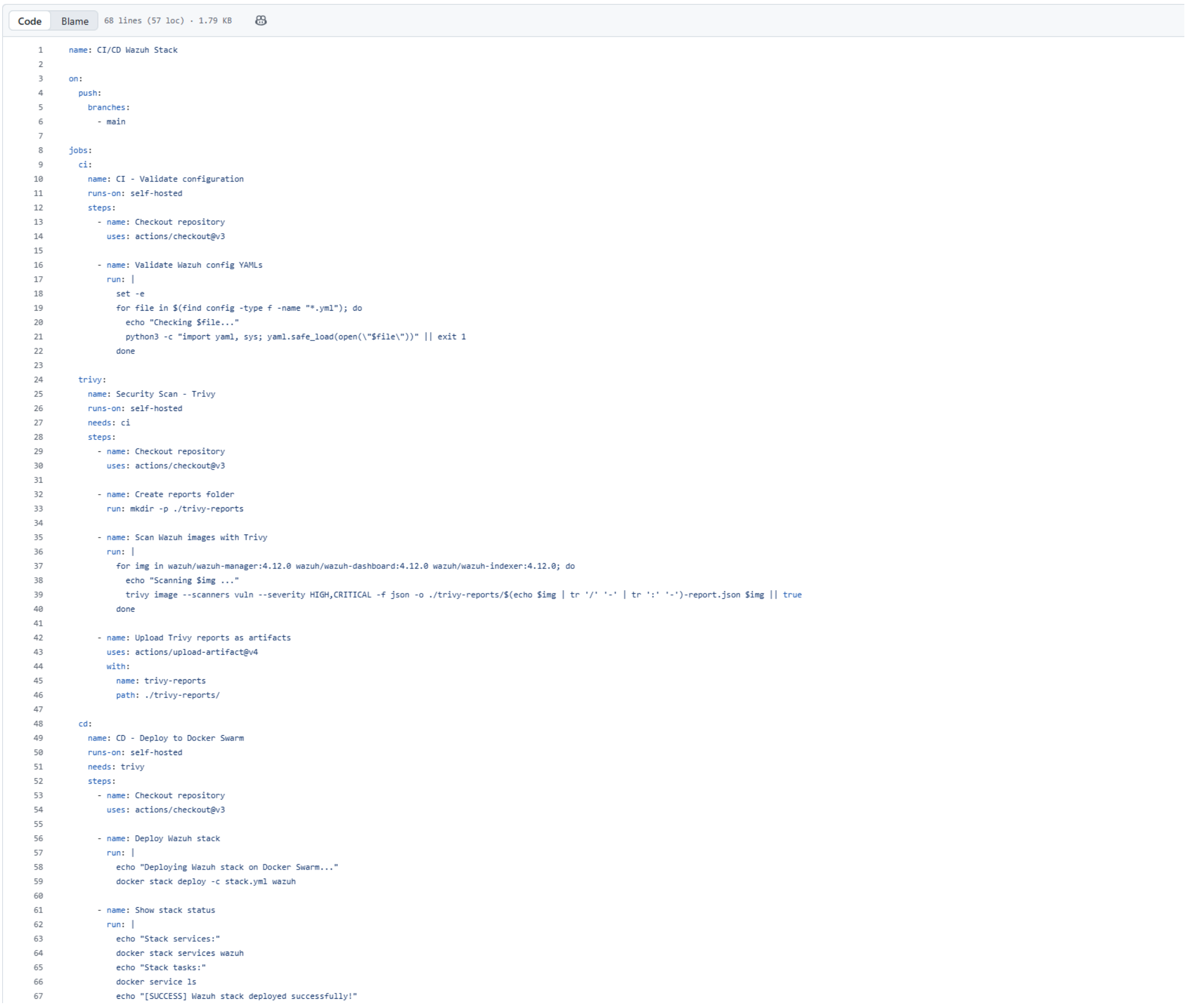
The core configuration for the entire Wazuh stack is managed within the config directory. This includes dedicated sub-folders for each component (e.g., wazuh_cluster, wazuh_indexer, wazuh_dashboard), ensuring a modular and organized setup. The SSL certificates required for secure communication between components are stored in wazuh_indexer_ssl_certs, while the custom threat detection rules for Part 2 of the challenge are located in wazuh_rules. The stack.yml file is the central deployment manifest, defining the services, networks, and volumes for the entire Wazuh architecture, enabling a single-command deployment to the Docker Swarm cluster.



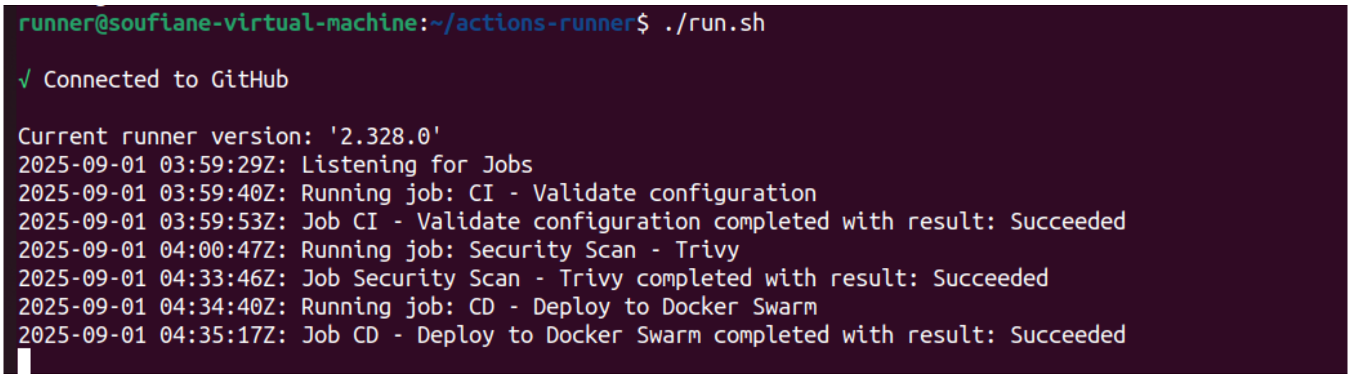
```
Wazuh_Stack/
├── config
│   ├── certs.yml
│   ├── nginx
│   │   └── nginx.conf
│   ├── wazuh_cluster
│   │   ├── wazuh_manager.conf
│   │   └── wazuh_worker.conf
│   ├── wazuh_dashboard
│   │   ├── opensearch_dashboards.yml
│   │   └── wazuh.yml
│   ├── wazuh_indexer
│   │   ├── internal_users.yml
│   │   ├── wazuh1.indexer.yml
│   │   ├── wazuh2.indexer.yml
│   │   └── wazuh3.indexer.yml
│   ├── wazuh_indexer_ssl_certs
│   │   ├── admin-key.pem
│   │   ├── admin.pem
│   │   ├── root-ca.key
│   │   ├── root-ca-manager.key
│   │   ├── root-ca-manager.pem
│   │   ├── root-ca.pem
│   │   ├── wazuh1.indexer-key.pem
│   │   ├── wazuh1.indexer.pem
│   │   ├── wazuh2.indexer-key.pem
│   │   ├── wazuh2.indexer.pem
│   │   ├── wazuh3.indexer-key.pem
│   │   ├── wazuh3.indexer.pem
│   │   ├── wazuh.dashboard-key.pem
│   │   ├── wazuh.dashboard.pem
│   │   ├── wazuh.master-key.pem
│   │   ├── wazuh.master.pem
│   │   ├── wazuh.worker-key.pem
│   │   └── wazuh.worker.pem
│   └── wazuh_rules
│       ├── 0010-custom_sshd.xml
│       ├── 0998-local_ssh_decoders.xml
│       └── 0999-local_ssh_rules.xml
```

CI/CD Pipeline and Automation

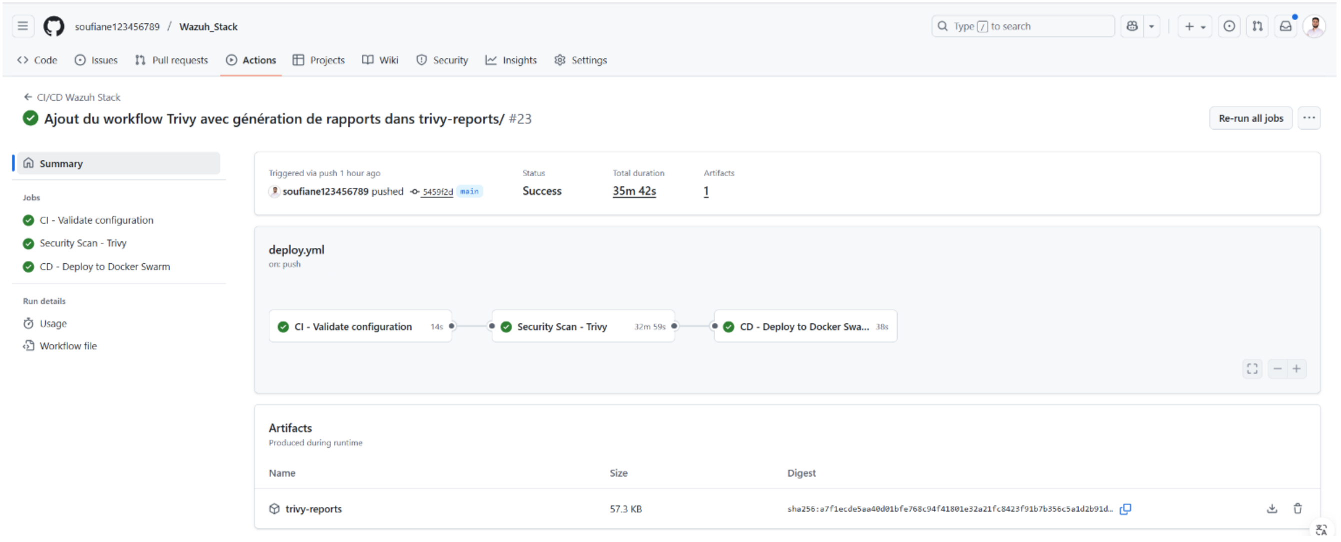
A custom CI/CD pipeline was implemented using GitHub Actions to automate the entire deployment workflow. The pipeline's initial stages focused on validating the YAML files and checking for syntax errors to ensure code integrity. This was followed by a critical security step where all Docker images within the stack were scanned for vulnerabilities using **Trivy**. To bridge the gap between GitHub and the on-premise virtual machine, a self-hosted GitHub Runner was installed on the VM. This allowed the pipeline to securely and directly deploy the entire Wazuh stack from the GitHub repository to the local machine, ensuring a seamless and automated deployment process.



```yaml
name: CI/CD Wazuh Stack

on:
  push:
    branches:
      - main

jobs:
  ci:
    name: CI - Validate configuration
    runs-on: self-hosted
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Validate Wazuh config YAMLs
        run: |
          set -e
          for file in $(find config -type f -name "*.yml"); do
            echo "Checking $file..."
            python3 -c "import yaml, sys; yaml.safe_load(open(\"$file\"))" || exit 1
          done

  trivy:
    name: Security Scan - Trivy
    runs-on: self-hosted
    needs: ci
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Create reports folder
        run: mkdir -p ./trivy-reports

      - name: Scan Wazuh images with Trivy
        run: |
          for img in wazuh/wazuh-manager:4.12.0 wazuh/wazuh-dashboard:4.12.0 wazuh/wazuh-indexer:4.12.0; do
            echo "Scanning $img ..."
            trivy image --scanners vuln --severity HIGH,CRITICAL -f json -o ./trivy-reports/$(echo $img | tr '/' '-' | tr ':' '-')-report.json $img || true
          done

      - name: Upload Trivy reports as artifacts
        uses: actions/upload-artifact@v4
        with:
          name: trivy-reports
          path: ./trivy-reports/

  cd:
    name: CD - Deploy to Docker Swarm
    runs-on: self-hosted
    needs: trivy
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Deploy Wazuh stack
        run: |
          echo "Deploying Wazuh stack on Docker Swarm..."
          docker stack deploy -c stack.yml wazuh

      - name: Show stack status
        run: |
          echo "Stack services:"
          docker stack services wazuh
          echo "Stack tasks:"
          docker service ls
          echo "[SUCCESS] Wazuh stack deployed successfully!"
```



```
runner@soufiane-virtual-machine:~/actions-runner$ ./run.sh

√ Connected to GitHub

Current runner version: '2.328.0'
2025-09-01 03:59:29Z: Listening for Jobs
2025-09-01 03:59:40Z: Running job: CI - Validate configuration
2025-09-01 03:59:53Z: Job CI - Validate configuration completed with result: Succeeded
2025-09-01 04:00:47Z: Running job: Security Scan - Trivy
2025-09-01 04:33:46Z: Job Security Scan - Trivy completed with result: Succeeded
2025-09-01 04:34:40Z: Running job: CD - Deploy to Docker Swarm
2025-09-01 04:35:17Z: Job CD - Deploy to Docker Swarm completed with result: Succeeded
```



soufiane123456789 / Wazuh_Stack

← CI/CD Wazuh Stack

Ajout du workflow Trivy avec génération de rapports dans trivy-reports/ #23

Jobs
- CI - Validate configuration
- Security Scan - Trivy
- CD - Deploy to Docker Swarm

Triggered via push 1 hour ago: soufiane123456789 pushed 5459f2d main
Status: Success
Total duration: 35m 42s
Artifacts: 1

deploy.yml
on: push

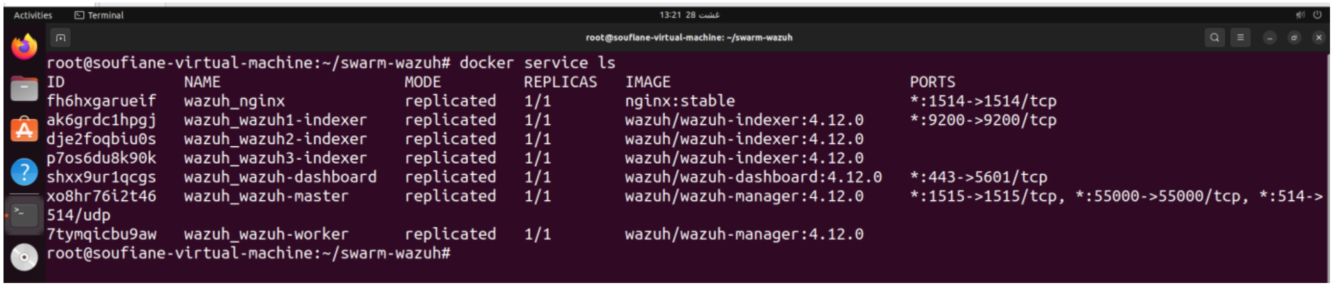
CI - Validate configuration 14s → Security Scan - Trivy 32m 59s → CD - Deploy to Docker Swa... 38s

Artifacts
Produced during runtime

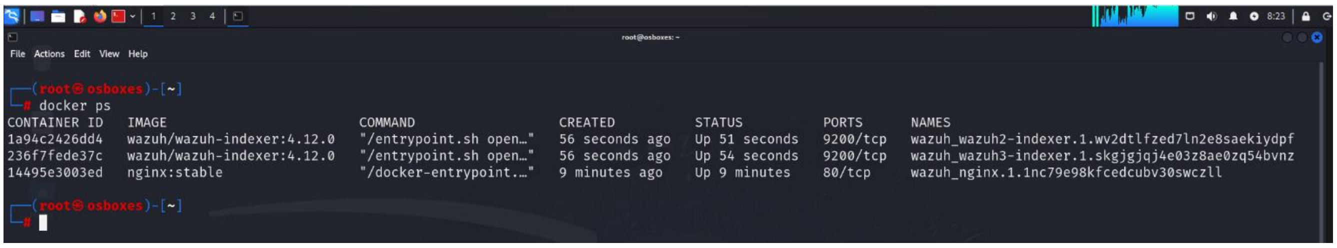
Name	Size	Digest
trivy-reports	57.3 KB	sha256:a7f1ecde5aa40d01bfe768c94f41801e32a21fc8423f91b7b356c5a1d2b91d...

Deployment Validation

Following the deployment process, it was confirmed that all Wazuh services were successfully deployed on our Swarm cluster. The containers were optimally distributed across the manager and worker nodes, thereby ensuring the high availability and resilience of the entire stack. This distribution validates the robustness of the architecture and the success of the automation.



```
root@soufiane-virtual-machine:~/swarm-wazuh# docker service ls
ID             NAME                     MODE         REPLICAS   IMAGE                             PORTS
fh6hxgarueif   wazuh_nginx              replicated   1/1        nginx:stable                      *:1514->1514/tcp
ak6grdc1hpgj   wazuh_wazuh1-indexer     replicated   1/1        wazuh/wazuh-indexer:4.12.0        *:9200->9200/tcp
dje2foqbiu0s   wazuh_wazuh2-indexer     replicated   1/1        wazuh/wazuh-indexer:4.12.0
p7os6du8k90k   wazuh_wazuh3-indexer     replicated   1/1        wazuh/wazuh-indexer:4.12.0
shxx9ur1qcgs   wazuh_wazuh-dashboard    replicated   1/1        wazuh/wazuh-dashboard:4.12.0      *:443->5601/tcp
xo8hr76i2t46   wazuh_wazuh-master       replicated   1/1        wazuh/wazuh-manager:4.12.0        *:1515->1515/tcp, *:55000->55000/tcp, *:514->
514/udp
7tymqicbu9aw   wazuh_wazuh-worker       replicated   1/1        wazuh/wazuh-manager:4.12.0
root@soufiane-virtual-machine:~/swarm-wazuh#
```



```
┌──(root㉿osboxes)-[~]
└─# docker ps
CONTAINER ID   IMAGE                        COMMAND                  CREATED          STATUS          PORTS      NAMES
1a94c2426dd4   wazuh/wazuh-indexer:4.12.0   "/entrypoint.sh open…"   56 seconds ago   Up 51 seconds   9200/tcp   wazuh_wazuh2-indexer.1.wv2dtlfzed7ln2e8saekiydpf
236f7fede37c   wazuh/wazuh-indexer:4.12.0   "/entrypoint.sh open…"   56 seconds ago   Up 54 seconds   9200/tcp   wazuh_wazuh3-indexer.1.skgjgjqj4e03z8ae0zq54bvnz
14495e3003ed   nginx:stable                 "/docker-entrypoint.…"   9 minutes ago    Up 9 minutes    80/tcp     wazuh_nginx.1.1nc79e98kfcedcubv30swczll

┌──(root㉿osboxes)-[~]
└─#
```

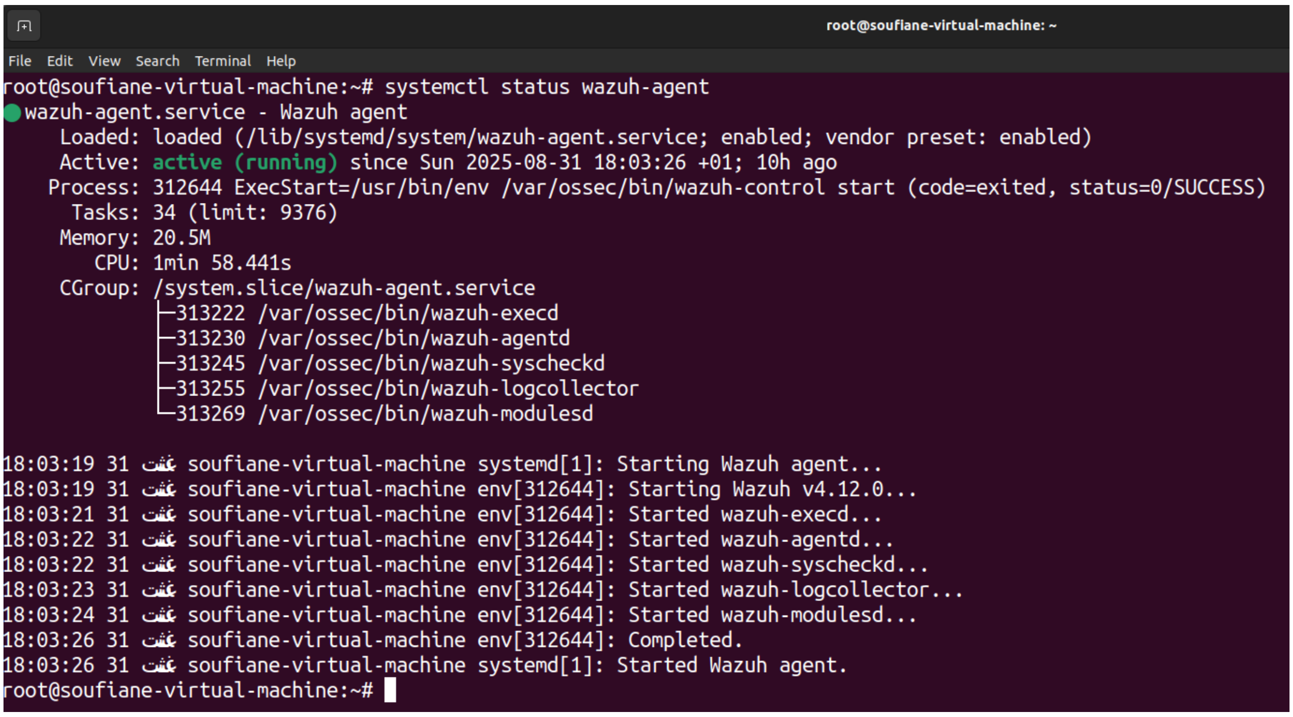
Part 2: Custom Threat Detection Rule

Introduction

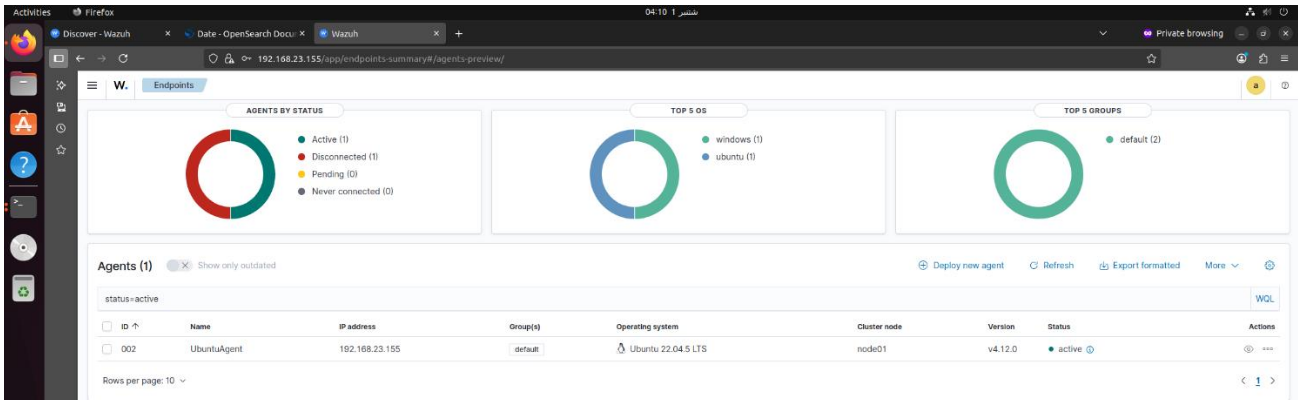
The second part of this project focused on enhancing the security monitoring capabilities of the Wazuh stack by implementing a custom detection rule. This rule was specifically designed to identify suspicious SSH login patterns, a common vector for brute-force attacks and unauthorized access attempts. The objective was to create an automated alert system that detects a sequence of multiple failed login attempts followed by a successful login with a new user, providing immediate visibility into potential security incidents.

Wazuh Agent Installation

To monitor the target node, a Wazuh agent was successfully downloaded, installed, and activated on the Ubuntu machine. Once connected to the Wazuh server, the agent began collecting security logs and forwarding them for analysis and detection, serving as the data source for our custom rule.


root@soufiane-virtual-machine: ~
File Edit View Search Terminal Help

```
root@soufiane-virtual-machine:~# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-08-31 18:03:26 +01; 10h ago
    Process: 312644 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 34 (limit: 9376)
     Memory: 20.5M
        CPU: 1min 58.441s
     CGroup: /system.slice/wazuh-agent.service
             ├─313222 /var/ossec/bin/wazuh-execd
             ├─313230 /var/ossec/bin/wazuh-agentd
             ├─313245 /var/ossec/bin/wazuh-syscheckd
             ├─313255 /var/ossec/bin/wazuh-logcollector
             └─313269 /var/ossec/bin/wazuh-modulesd

18:03:19 31 غشت soufiane-virtual-machine systemd[1]: Starting Wazuh agent...
18:03:19 31 غشت soufiane-virtual-machine env[312644]: Starting Wazuh v4.12.0...
18:03:21 31 غشت soufiane-virtual-machine env[312644]: Started wazuh-execd...
18:03:22 31 غشت soufiane-virtual-machine env[312644]: Started wazuh-agentd...
18:03:22 31 غشت soufiane-virtual-machine env[312644]: Started wazuh-syscheckd...
18:03:23 31 غشت soufiane-virtual-machine env[312644]: Started wazuh-logcollector...
18:03:24 31 غشت soufiane-virtual-machine env[312644]: Started wazuh-modulesd...
18:03:26 31 غشت soufiane-virtual-machine env[312644]: Completed.
18:03:26 31 غشت soufiane-virtual-machine systemd[1]: Started Wazuh agent.
root@soufiane-virtual-machine:~# 
```

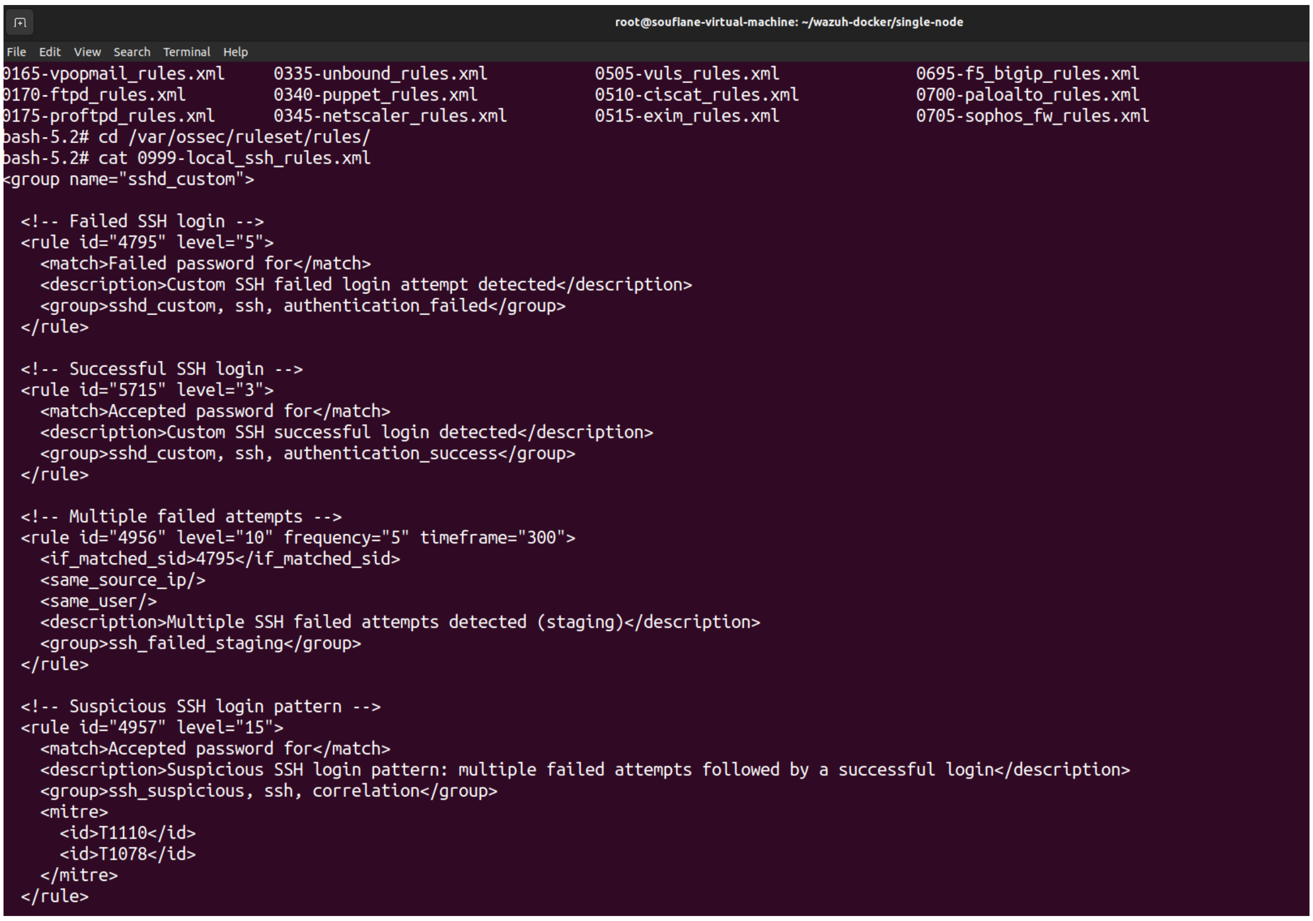


status=active

ID ↑	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
002	UbuntuAgent	192.168.23.155	default	Ubuntu 22.04.5 LTS	node01	v4.12.0	● active	

Custom Rule Implementation

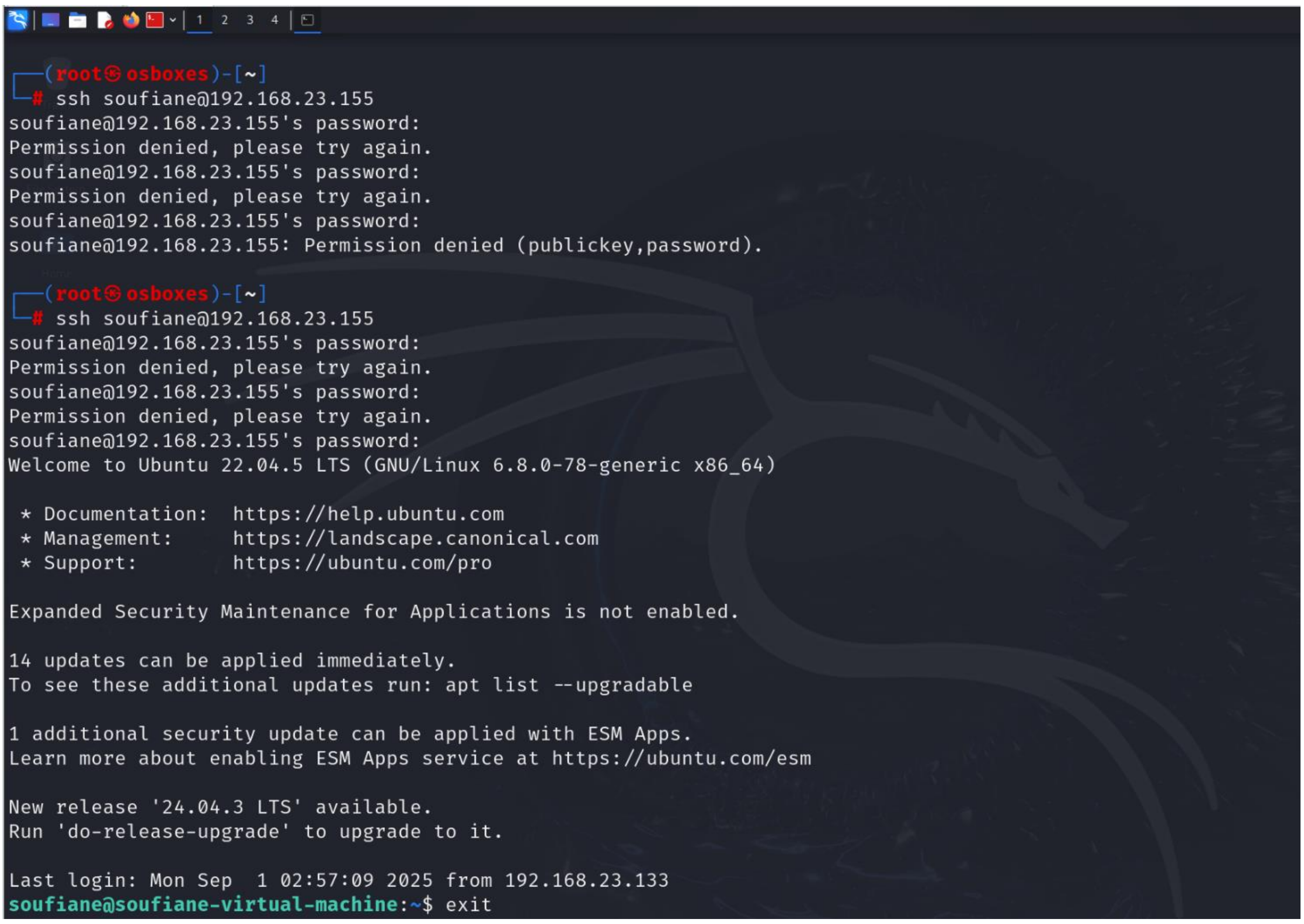
To create the custom rule, we accessed the Wazuh Manager container's shell and navigated to the rules directory at /var/ossec/rulesets/rules. Within this location, a new rule was created to identify suspicious SSH login activity. The logic of the rule is specifically designed to detect a sequence of five failed SSH login attempts from the same IP address. Following these failures, if a successful login is registered from the same IP and with the same user, a "suspicious SSH login" alert is generated. This custom rule directly addresses the threat detection requirements of the challenge, providing targeted and effective security monitoring.


root@soufiane-virtual-machine: ~/wazuh-docker/single-node
File Edit View Search Terminal Help

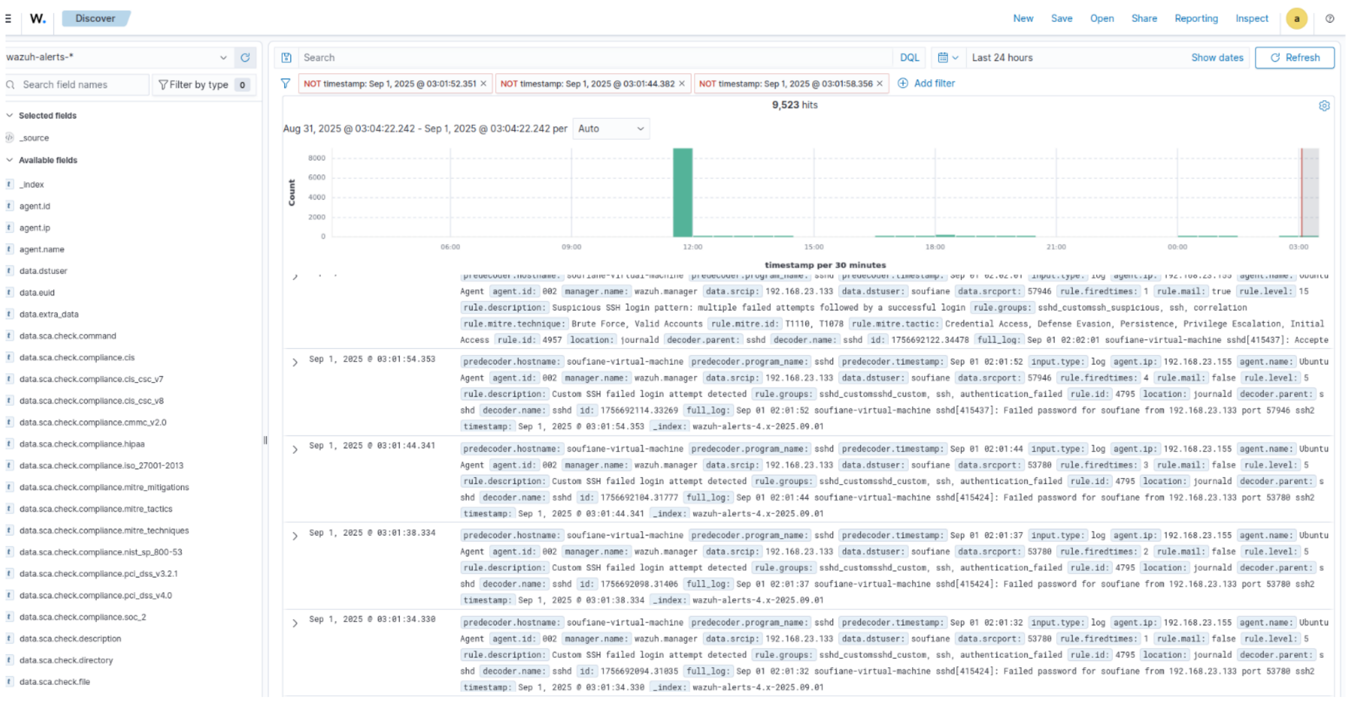
```
0165-vpopmail_rules.xml     0335-unbound_rules.xml          0505-vuls_rules.xml             0695-f5_bigip_rules.xml
0170-ftpd_rules.xml         0340-puppet_rules.xml           0510-ciscat_rules.xml           0700-paloalto_rules.xml
0175-proftpd_rules.xml      0345-netscaler_rules.xml        0515-exim_rules.xml             0705-sophos_fw_rules.xml
bash-5.2# cd /var/ossec/ruleset/rules/
bash-5.2# cat 0999-local_ssh_rules.xml
<group name="sshd_custom">

  <!-- Failed SSH login -->
  <rule id="4795" level="5">
    <match>Failed password for</match>
    <description>Custom SSH failed login attempt detected</description>
    <group>sshd_custom, ssh, authentication_failed</group>
  </rule>

  <!-- Successful SSH login -->
  <rule id="5715" level="3">
    <match>Accepted password for</match>
    <description>Custom SSH successful login detected</description>
    <group>sshd_custom, ssh, authentication_success</group>
  </rule>

  <!-- Multiple failed attempts -->
  <rule id="4956" level="10" frequency="5" timeframe="300">
    <if_matched_sid>4795</if_matched_sid>
    <same_source_ip/>
    <same_user/>
    <description>Multiple SSH failed attempts detected (staging)</description>
    <group>ssh_failed_staging</group>
  </rule>

  <!-- Suspicious SSH login pattern -->
  <rule id="4957" level="15">
    <match>Accepted password for</match>
    <description>Suspicious SSH login pattern: multiple failed attempts followed by a successful login</description>
    <group>ssh_suspicious, ssh, correlation</group>
    <mitre>
      <id>T1110</id>
      <id>T1078</id>
    </mitre>
  </rule>
```

Alert Triggering

Following the implementation of the custom rule, a simulation was performed from a Kali Linux machine. After five unsuccessful SSH login attempts, the sixth attempt, with the correct password, succeeded. As designed, this sequence of events immediately triggered an alert. Screenshots of the generated alerts, visible in the Wazuh dashboard, confirm that our rule successfully detected the suspicious login pattern, thereby validating its proper functioning.



```
┌──(root㉿osboxes)-[~]
└─# ssh soufiane@192.168.23.155
soufiane@192.168.23.155's password:
Permission denied, please try again.
soufiane@192.168.23.155's password:
Permission denied, please try again.
soufiane@192.168.23.155's password:
soufiane@192.168.23.155: Permission denied (publickey,password).

┌──(root㉿osboxes)-[~]
└─# ssh soufiane@192.168.23.155
soufiane@192.168.23.155's password:
Permission denied, please try again.
soufiane@192.168.23.155's password:
Permission denied, please try again.
soufiane@192.168.23.155's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-78-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

14 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Sep  1 02:57:09 2025 from 192.168.23.133
soufiane@soufiane-virtual-machine:~$ exit
```

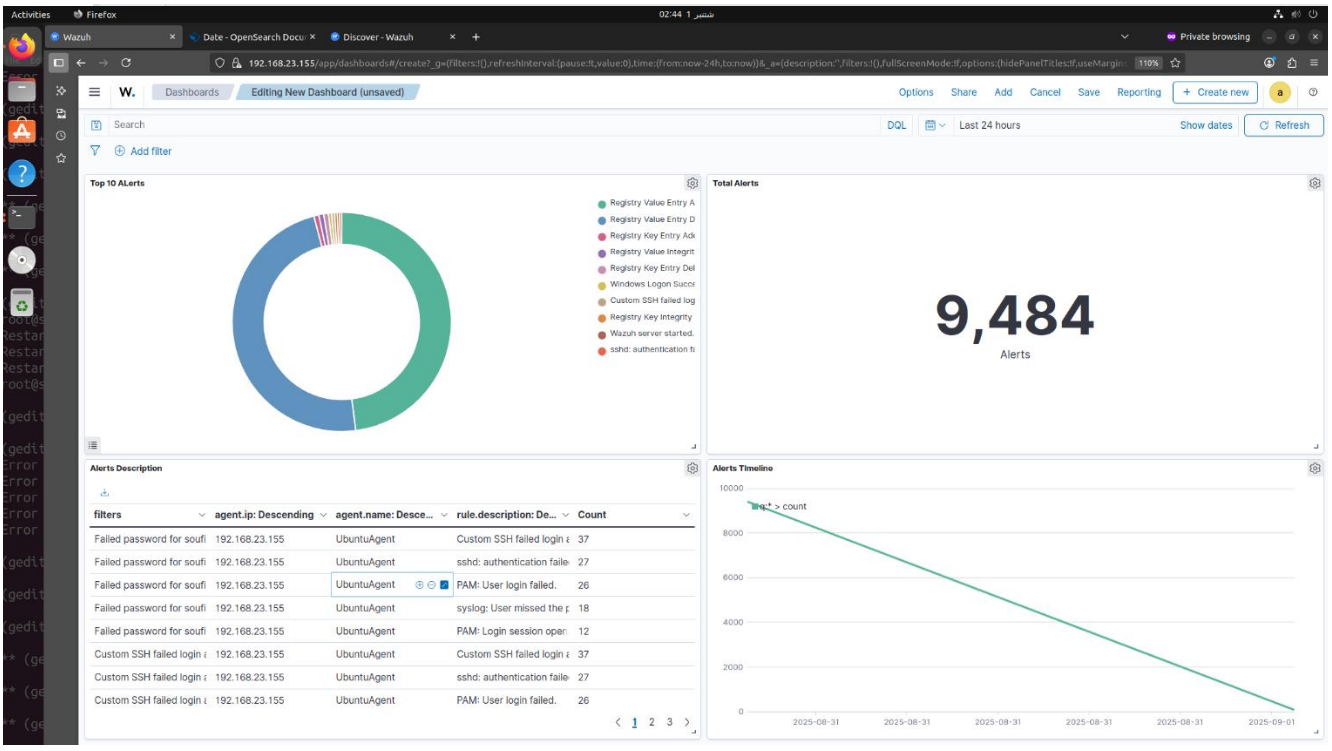


Discover
New
Save
Open
Share
Reporting
Inspect
wazuh-alerts-*
Search field names
Filter by type 0
Selected fields
_source
Available fields
_index
agent.id
agent.ip
agent.name
data.dstuser
data.euid
data.extra_data
data.sca.check.command
data.sca.check.compliance.cis
data.sca.check.compliance.cis_csc_v7
data.sca.check.compliance.cis_csc_v8
data.sca.check.compliance.cmmc_v2.0
data.sca.check.compliance.hipaa
data.sca.check.compliance.iso_27001-2013
data.sca.check.compliance.mitre_mitigations
data.sca.check.compliance.mitre_tactics
data.sca.check.compliance.mitre_techniques
data.sca.check.compliance.nist_sp_800-53
data.sca.check.compliance.pci_dss_v3.2.1
data.sca.check.compliance.pci_dss_v4.0
data.sca.check.compliance.soc_2
data.sca.check.description
data.sca.check.directory
data.sca.check.file
Search
DQL
Last 24 hours
Show dates
Refresh
NOT timestamp: Sep 1, 2025 @ 03:01:52.351
NOT timestamp: Sep 1, 2025 @ 03:01:44.382
NOT timestamp: Sep 1, 2025 @ 03:01:58.356
Add filter
9,523 hits
Aug 31, 2025 @ 03:04:22.242 - Sep 1, 2025 @ 03:04:22.242 per Auto
Count
timestamp per 30 minutes

```
Agent agent.id: 002 manager.name: wazuh.manager data.srcip: 192.168.23.133 data.dstuser: soufiane data.srcport: 57946 rule.firedtimes: 1 rule.mail: true rule.level: 15
rule.description: Suspicious SSH login pattern: multiple failed attempts followed by a successful login rule.groups: sshd_customssh_suspicious, ssh, correlation
rule.mitre.technique: Brute Force, Valid Accounts rule.mitre.id: T1110, T1078 rule.mitre.tactic: Credential Access, Defense Evasion, Persistence, Privilege Escalation, Initial
Access rule.id: 4957 location: journald decoder.parent: sshd decoder.name: sshd id: 1756692122.34478 full_log: Sep 01 02:02:01 soufiane-virtual-machine sshd[415437]: Accepte

> Sep 1, 2025 @ 03:01:54.353   predecoder.hostname: soufiane-virtual-machine predecoder.program_name: sshd predecoder.timestamp: Sep 01 02:01:52 input.type: log agent.ip: 192.168.23.155 agent.name: Ubuntu
Agent agent.id: 002 manager.name: wazuh.manager data.srcip: 192.168.23.133 data.dstuser: soufiane data.srcport: 57946 rule.firedtimes: 4 rule.mail: false rule.level: 5
rule.description: Custom SSH failed login attempt detected rule.groups: sshd_customsshd_custom, ssh, authentication_failed rule.id: 4795 location: journald decoder.parent: s
shd decoder.name: sshd id: 1756692114.33269 full_log: Sep 01 02:01:52 soufiane-virtual-machine sshd[415437]: Failed password for soufiane from 192.168.23.133 port 57946 ssh2
timestamp: Sep 1, 2025 @ 03:01:54.353 _index: wazuh-alerts-4.x-2025.09.01

> Sep 1, 2025 @ 03:01:44.341   predecoder.hostname: soufiane-virtual-machine predecoder.program_name: sshd predecoder.timestamp: Sep 01 02:01:44 input.type: log agent.ip: 192.168.23.155 agent.name: Ubuntu
Agent agent.id: 002 manager.name: wazuh.manager data.srcip: 192.168.23.133 data.dstuser: soufiane data.srcport: 53780 rule.firedtimes: 3 rule.mail: false rule.level: 5
rule.description: Custom SSH failed login attempt detected rule.groups: sshd_customsshd_custom, ssh, authentication_failed rule.id: 4795 location: journald decoder.parent: s
shd decoder.name: sshd id: 1756692104.31777 full_log: Sep 01 02:01:44 soufiane-virtual-machine sshd[415424]: Failed password for soufiane from 192.168.23.133 port 53780 ssh2
timestamp: Sep 1, 2025 @ 03:01:44.341 _index: wazuh-alerts-4.x-2025.09.01

> Sep 1, 2025 @ 03:01:38.334   predecoder.hostname: soufiane-virtual-machine predecoder.program_name: sshd predecoder.timestamp: Sep 01 02:01:37 input.type: log agent.ip: 192.168.23.155 agent.name: Ubuntu
Agent agent.id: 002 manager.name: wazuh.manager data.srcip: 192.168.23.133 data.dstuser: soufiane data.srcport: 53780 rule.firedtimes: 2 rule.mail: false rule.level: 5
rule.description: Custom SSH failed login attempt detected rule.groups: sshd_customsshd_custom, ssh, authentication_failed rule.id: 4795 location: journald decoder.parent: s
shd decoder.name: sshd id: 1756692098.31406 full_log: Sep 01 02:01:37 soufiane-virtual-machine sshd[415424]: Failed password for soufiane from 192.168.23.133 port 53780 ssh2
timestamp: Sep 1, 2025 @ 03:01:38.334 _index: wazuh-alerts-4.x-2025.09.01

> Sep 1, 2025 @ 03:01:34.330   predecoder.hostname: soufiane-virtual-machine predecoder.program_name: sshd predecoder.timestamp: Sep 01 02:01:32 input.type: log agent.ip: 192.168.23.155 agent.name: Ubuntu
Agent agent.id: 002 manager.name: wazuh.manager data.srcip: 192.168.23.133 data.dstuser: soufiane data.srcport: 53780 rule.firedtimes: 1 rule.mail: false rule.level: 5
rule.description: Custom SSH failed login attempt detected rule.groups: sshd_customsshd_custom, ssh, authentication_failed rule.id: 4795 location: journald decoder.parent: s
shd decoder.name: sshd id: 1756692094.31035 full_log: Sep 01 02:01:32 soufiane-virtual-machine sshd[415424]: Failed password for soufiane from 192.168.23.133 port 53780 ssh2
timestamp: Sep 1, 2025 @ 03:01:34.330 _index: wazuh-alerts-4.x-2025.09.01
```

Custom Dashboard and Visualization

Following the successful generation of alerts and logs, a custom dashboard was created to enhance the visualization of the security data. This tailored dashboard provides a clear and intuitive overview of the suspicious activities, including charts and graphs that represent the failed SSH login attempts and the successful intrusion. This visual representation allows for a more effective analysis of the events, enabling security professionals to quickly identify trends, patterns, and potential threats.



filters	agent.ip: Descending	agent.name: Desce...	rule.description: De...	Count
Failed password for soufi	192.168.23.155	UbuntuAgent	Custom SSH failed login a	37
Failed password for soufi	192.168.23.155	UbuntuAgent	sshd: authentication faile	27
Failed password for soufi	192.168.23.155	UbuntuAgent	PAM: User login failed.	26
Failed password for soufi	192.168.23.155	UbuntuAgent	syslog: User missed the p	18
Failed password for soufi	192.168.23.155	UbuntuAgent	PAM: Login session open	12
Custom SSH failed login a	192.168.23.155	UbuntuAgent	Custom SSH failed login a	37
Custom SSH failed login a	192.168.23.155	UbuntuAgent	sshd: authentication faile	27
Custom SSH failed login a	192.168.23.155	UbuntuAgent	PAM: User login failed.	26